APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: **METHOD AND APPARATUS FOR MANAGING SYSTEM RESOURCES USING A CONTAINER MODEL**

APPLICANTS: **Ling L. CHEN, Joost W. PRONK VAN HOOGEVEEN, Nils P. PEDERSEN, Xuemei GAO, Amol A. CHIPLUNKAR, Harapanahalli Gururaja RAO, John M. PERRY, Ross A. MARGO, and Estelle J. CHIQUET**

"EXPRESS MAIL" Mailing Label Number: EV436028465US
Date of Deposit: April 14, 2004

METHOD AND APPARATUS FOR MANAGING SYSTEM RESOURCES USING A CONTAINER MODEL

Background

[0001] As the complexity of computer systems has increased, the operating system executing in the computer systems have also become more sophisticated. Specifically, operating systems have been augmented to include resource management utilities. Resource management utilities enable system users to control how processes executing on the computer system use the available system resources (*e.g.,* central processing units (CPUs), physical memory, bandwidth, etc.).

[0002] The resource management utilities provide a number of different resource management control mechanisms to control system resource allocation for processes, such as CPU shares allocation, physical memory control, system resource partitioning, etc. Each of the aforementioned resource management control mechanisms may be applied independently and individually to each application (or service (*i.e.,* group of related applications) executing on the computer system. Alternatively, each of the aforementioned resource management control mechanisms may be applied to one or more applications in combination. The resource management control mechanisms described above may be generically categorized into three distinct types: constraint mechanisms, scheduling mechanisms, and partitioning mechanisms.

[0003] Constraint mechanisms allow a system user (*e.g.,* a system administrator) or an application developer to set bounds on the consumption of a specific system resource for an application or one or more processes within an application. Once the bounds on resource consumption are specified, the system user and/or application developer may readily and easily model system resource consumption scenarios. Further, the bounds on resource consumption may also be used to control ill-behaved applications that would otherwise compromise computer system performance or availability through unregulated system resource requests.

[0004] Scheduling mechanisms allow a system to make a sequence of system resource allocation decisions at specific time intervals. The decision about how to allocate a particular system resource is typically based on a predictable algorithm. In some instances, the scheduling algorithm might guarantee that all applications have some access to the system resource. Scheduling mechanisms enable improved utilization of an under committed system resource, while providing controlled allocations in a critically committed or overcommitted system resource utilization scenario.

[0005] Partitioning mechanisms allow a system user to bind a particular application (or one or more sub-processes within the application) to a subset of available system resources. This binding guarantees that a known amount of system resources is always available to the application (or one or more sub-processes within the application).

[0006] Operating systems providing the above-mentioned resource management control mechanisms typically provide a resource management utility and/or an application programming interface (API) for each of the mechanisms. Further, each of the resource management utilities/APIs provides a number of different options for the system user to select and configure. In general, the resource management utilities/APIs are not tightly coupled together. In addition, as the operating system evolves, different versions of the resource management utilities/APIs become available that have different options and functionality. Accordingly, in order for a system user to take advantage of the aforementioned mechanisms, the system user typically requires a thorough understanding of the different types of resource management control mechanisms as well as the relationship/interaction between the various resource management control mechanisms.

Summary

[0007] In general, in one aspect, the invention relates to a A method for managing system resources, comprising creating a container, wherein creating the container comprises allocating a first portion of a first resource to the container, associating the container with a resource pool, wherein the resource pool is associated with the first resource, determining whether the first portion of the first resource is valid, and activating the container if the first portion of the first resource is valid.

[0008] In general, in one aspect, the invention relates to a resource management system, comprising a first resource and a second resource, a first resource pool, wherein the resource pool is allocated a portion of the first resource and a portion of the second resource, a first container residing in the first resource pool, wherein the first container comprises a requirements specification for the first resource for the first container and a requirements specification for the second resource for the first container, and a management interface configured to verify the requirements specification for the first resource with the allocated portion of the first resource, and verify the requirements specification for the second resource with the allocated portion of the second resource.

[0009] In general, in one aspect, the invention relates to a network system having a plurality of nodes, including a first resource and a second resource, a first resource pool, wherein the resource pool is allocated a portion of the first resource and a portion of the second resource, a first container residing in the first resource pool, wherein the first container comprises a requirements specification for the first resource for the first container and a requirements specification for the second resource for the first container, and a management interface configured to verify the requirements specification for the first resource with the allocated portion of the first resource, and verify the requirements specification for the second resource with the allocated portion of the second resource, wherein the first resource is located on any one of the plurality of nodes, wherein the second resource is located on any one of the plurality of nodes, wherein the first resource pool is located on any one of the plurality of nodes, wherein the container is located on any one of the plurality of nodes, wherein the management interface executes on any one of the plurality of nodes.

[0010] Other aspects of the invention will be apparent from the following description and the appended claims.

Brief Description of Drawings

[0011] Figure 1 shows a system architecture in accordance with one embodiment of the invention.

[0012] Figure 2 shows a view of a container level in accordance with one embodiment of the system.

[0013] Figure 3 shows a view of a resource pool in accordance with one embodiment of the invention.

[0014] Figure 4 shows a flow chart in accordance with one embodiment of the invention.

[0015] Figure 5 shows a computer system in accordance with one embodiment of the invention.

Detailed Description

[0016] Exemplary embodiments of the invention will be described with reference to the accompanying drawings. Like items in the drawings are shown with the same reference numbers.

[0017] In one or more embodiments of the invention, numerous specific details are set forth in order to provide a more thorough understanding of the invention. However, it will be apparent to one of ordinary skill in the art that the invention may be practiced without these specific details. In other instances, well-known features have not been described in detail to avoid obscuring the invention.

[0018] In general, embodiments of the invention provide a method and apparatus for system resource management. One or more embodiments of the invention provide a simple model that helps manage the complexity of the resource management control mechanisms and their different resource management utilities/APIs for system users. In one embodiment of the invention, a system user may create a container and define the system resources allocated to the particular container. Applications (and services) may then be executed within the container and their performance/system resource usage may be controlled and monitored.

[0019] Figure 1 shows a system architecture in accordance with one embodiment of the invention. The system includes hardware (100) (*e.g.,* CPUs, physical memory, network interface cards, etc.) and an operating system (102) executing on the hardware (100).

The operating system (102) includes a number of management utilities (*i.e.*, Management Utility A (104A), Management Utility B (104B), Management Utility C (104C)). The management utilities (105) (*i.e.*, Management Utility A (104A), Management Utility B (104B), Management Utility C (104C)) typically provide resource management control mechanisms (*e.g.*, constraint mechanisms, scheduling mechanisms, partitioning mechanisms, etc.) to manage system resources (*e.g.*, CPUs, physical memory, bandwidth, etc.).

[0020] In one embodiment of the invention, a container level (106) interfaces with the management utilities (*i.e.*, Management Utility A (104A), Management Utility B (104B), Management Utility C (104C)). The container level (106) typically includes one or more containers. The container level (106) is described in Figures 2 and 3. The container level (106) interfaces with a management interface (108). The management interface (108) provides a system user an interface to specify how resources are to be allocated within a given container in the container level (106). In one embodiment of the invention, the management interface (108) includes a graphical user interface to allow the system user to specify the system resource allocation.

[0021] In one embodiment of the invention, the management interface (108) may include functionality to define a container, create a resource pool (as shown in Figure 2), deploy the container in a particular resource pool, activate the container, modify the container (*e.g.*, modify the resources allocated to the container), deactivate the container, and delete the container.

[0022] As noted above, the management interface includes functionality to modify the container definition for one or more containers. In one embodiment of the invention, this functionality is extended to allow a system user to change the container definition for deployed containers that have the same container definition. The aforementioned functionality is hereafter referred to as a "Schedule Change `" functionality.

[0023] In one embodiment of the invention, the Schedule Change Job functionality is implemented in the following manner. Initially, a system user creates a job on selected deployed containers that have the same container definition. Once the job is created it

may be executed immediately or at a scheduled time. The changes specified in the job will be applied to those selected containers. In one embodiment of the invention, the job is specific to a container, and may include a list of hosts on which the containers are deployed, a resource specification for one or more system resources, and a schedule to run the job.

[0024] Consider the following example, in an enterprise environment, a container called Web Service, which is created for all Web Service related processes, is deployed on a system located in San Francisco and on another system in New York City. During the business hours (peak time), the Web Service containers require 4 CPUs and 2GB of physical memory to handle all requests. In the evening, the Web Service containers only require 1 CPU and 512MB of physical memory.

[0025] In this scenario, the system administrator could create the following two jobs to change the container definition for the Web Service Container. Specifically, Job 1 would set the CPU specification to 4 CPUs, the physical memory cap to 2GB, and would be scheduled to execute at the beginning of business hours in the respective geographic locations. Job 2 would set the CPU specification to 1 CPU, the physical memory cap to 512MB, and would be scheduled to execute at the end of business hours in the respective geographic locations.

[0026] Further, the management interface (108) may include functionality to track system resource usage for a given container and graphically display the system resource usage of the particular container to the system user. In one embodiment of the invention, the management resource interface (108) may also include functionality to export a container's system resource usage in a format (*e.g.,* comma separated variable format) that may be imported into a word/accounting information processing tool. In addition, the management interface (108) may include functionality to determine whether a particular container or application/service running in the container is using more system resources than are allocated to the particular container and to notify/alert the system user if this situations exists. For example, the system user may be alerted via an e-mail message.

[0027] In one embodiment of the invention, the management interface includes functionality to identify, monitor, and measure the performance of two or more instances of a container (*i.e.*, distinct containers having the same container definition (described below)) executing on different systems on a network.

[0028] Further, the management interface (108) may include functionality to convert the system resource allocations specified by the system user for the containers into commands that are understood by the management utilities (105) (*i.e.*, Management Utility A (104A), Management Utility B (104B), Management Utility C (104C)). Further, the management interface (108) interfaces with a database (110). In one embodiment of the invention, the management interface (108) includes functionality to discover the system resources that are available on the particular computer system and store this information in the database (110).

[0029] In one embodiment of the invention, the database (110) stores the container definitions (*i.e.*, what system resources are allocated to the particular container, etc.) Further, the database (110) stores information about the system resources that are available on the computer system. In addition, the database (110) may include functionality to store the system resource usage per container.

[0030] Figure 2 shows a detailed view of a container level in accordance with one embodiment of the system. The container level (106) is partitioned into resource pools (*e.g.*, Resource Pool A (120) and Resource Pool B (122)). In one embodiment of the invention, the resource pools (*e.g.*, Resource Pool A (120) and Resource Pool B (122)) provide a persistent configuration mechanism for processor set configuration, and optionally, scheduling class assignment (*e.g.*, Fair Share Scheduling (FSS), Time Share Scheduling (TSS), etc.). Accordingly, each resource pool (*e.g.*, Resource Pool A (120) and Resource Pool B (122)) is associated with a processor set (*e.g.*, typically 2^n CPUs where n is a whole number).

[0031] In one embodiment of the invention, the following information may be stored in the database (110 in Figure 1) or an alternate location, for each resource pool: resource pool name, number of CPUs in the resource pool, the processor set associated with the

resource pool, the scheduling class assignment for the resource pool, etc. In one embodiment of the invention, the following information may be stored in the database (110 in Figure 1), or an alternate location, for each processor set: processor set name, processor set ID, number of CPUs in processor set, minimum size of processor set, maximum size of processor set, etc.

[0032] Residing in each resource pool (*e.g.*, Resource Pool A (120) and Resource Pool B (122)) is one or more containers. For example, Container A (124) and Container B (126) reside in resource pool A (120), while Container C (128), Container D (130), Container E (132), Container F (134), and Container G (136) reside in Resource Pool B (122). In one embodiment of the invention, each container may be described as a multi-dimensional system resource space in which an application (or service) may be executed. The size of the container is typically determined by the amount of each system resource allocated to the container (*e.g.*, the number of CPUs allocated, the amount of physical memory allocated, the amount of network bandwidth, etc).

[0033] Though not shown in Figure 2, applications (or services) executing in a container correspond to one or more processes. In one embodiment of the invention, all processes executing in a container are associated with the same ID (*e.g.*, a project ID). The ID may be used to track system resource usage for each container and also to enforce constraints on system resource usage.

[0034] In one embodiment of the invention, the following information may be stored in the database (110), or an alternate location, for each container: container ID, container name, project ID, description (*e.g.*, brief description of the properties of the container or the textual description used to identify the container, etc.), name of associated resource pool, minimum number of CPUs required for container activation, the minimum and/or maximum upload (*i.e.*, outgoing) bandwidth required, the minimum download bandwidth required, maximum physical memory allocated to the container, etc. In addition, the following information may be stored in the database, or an alternate location, for each container: real time CPU usage, CPU usage in percentage of total system, real time virtual memory usage, access control lists (ACL), and expressions.

[0035] In one embodiment of the invention, an ACL may be used to specify which users may be allowed to join the container and execute applications (or services) within the container. Further, the ACL may also specify groups of users that may join the container and execute applications (or services) within the container. In one embodiment, expressions correspond to regular expressions that may be used to determine whether a particular process may be moved into the container.

[0036] Figure 3 shows a detailed view of a resource pool in accordance with one embodiment of the invention. Allocated Resource A (146), Allocated Resource B (148), and Allocated Resource C (150) are allocated to Resource Pool A (120). In one embodiment of the invention, Allocated Resource A (146), Allocated Resource B (148), and Allocated Resource C (150) correspond to CPU, physical memory, and bandwidth, respectively. Further, each of the allocated resources (*e.g.,* Allocated Resource A (146), Allocated Resource B (148), and Allocated Resource C (150)) may individually correspond to the entire system resource available on the computer system or only a portion thereof.

[0037] The allocated resources (*e.g.,* Allocated Resource A (146), Allocated Resource B (148), and Allocated Resource C (150)) are further sub-divided by the containers (*i.e.,* Container A (124) and Container B (126)). Specifically, when a container is created, a container definition (140, 142) is specified. The container definition (140, 142) defines the resource requirements for each of the resources on a per container basis. For example, Container A (124) includes a container definition (140) that defines Resource A Requirements for Container A (140A), Resource B Requirements for Container A (140B), and Resource C Requirements for Container A (140C). Further, Container B (126) includes a container definition (142) that defines Resource A Requirements for Container B (142A), Resource B Requirements for Container B (142B), and Resource C Requirements for Container B (142C). In one embodiment of the invention, prior to placing a container in a given resource pool, the management interface (108) (or a related process) verifies whether the resource pool has sufficient resources to support the container.

[0038] In one embodiment of the invention, the minimum CPUs required for a given container is specified as the number of CPUs. For example, the system user may specify that 0.5 CPUs are required by the container, where 0.5 CPUs corresponds to 50% of a given CPU resource. Note that by specifying the number of CPUs required by the container, the management interface may readily validate the allocation by obtaining the total number of CPUs available in the computer system (which is calculated from the total number of CPUs in the computer system less the number of CPUs reserved by other containers). This approach provides an efficient way to avoid over-booking (or over-allocation) of CPUs within the computer system. In one embodiment of the invention, physical memory and bandwidth required by the container are specified and validated in the same manner as CPUs.

[0039] In one embodiment of the invention, to aid in defining resource requirements for a particular container, the number of CPUs, available size of physical memory, and available bandwidth are calculated and displayed to the system user when the container is created. The aforementioned information may aid the system user in entering appropriate resource requirements. In one embodiment of the invention, the container definition may be modified at any time after the container is created.

[0040] Though not shown in Figure 3, in one embodiment of the invention, each resource pool (*e.g.,* Resource Pool A (120)) may include a default container. The default container corresponds to a container that may use all the system resources allocated to the corresponding resource pool in which the default container resides. Typically, the default container is used when a non-default container in the resource pool is deactivated but an application (or service) is still executing in the non-default container. In this scenario, the application (or service) executing in the non-default container is transferred to the default container to continue execution.

[0041] Figure 4 shows a flow chart showing a method in accordance with one embodiment of the invention. Initially, system resources are discovered (Step 100). In one embodiment of the invention, the discovered system resources (including corresponding properties such as number, size, location, etc.) are stored in a database.

One or more resource pools are subsequently created (Step 102). Creating the resource pool typically includes associating a processor set and, optionally, a scheduling class assignment with the resource pool. Once the resource pool has been created, system resources may also be allocated to the resource pool (*e.g.*, CPUs, physical memory, bandwidth, etc.) (Step 104).

[0042] Once the resource pool has been created and system resources have been allocated to the resource pool, one or more containers may be created (Step 106). In one embodiment of the invention, creating the container includes generating a container definition in which individual system resource requirements are specified. In one embodiment of the invention, the container is created and resource requirements are specified using a container template.

[0043] Once the containers have been created and the resource requirements specified, the containers are placed within a resource pool (*i.e.*, a container is deployed) (Step 108). Prior to activating the container in the resource pool, the resource requirements for the container are validated to determine whether the resource pool in which the container is deployed contains the necessary resources to support the container (Step 110). If the resource requirements for the container are valid, the container is activated in the resource pool (Step 112). Once a container has been activated, the system users (or groups) as defined by the ACL specified for the container may execute applications (or services) within the container.

[0044] Alternatively, the resource requirements are modified and the container is re-validated until the resource pool can support the container (not shown). The container may also be deployed in another resource pool that includes sufficient system resources. Those skilled in the art will appreciate that the aforementioned steps do not necessarily need to be performed in the order shown in Figure 4.

[0045] In one embodiment of the invention, the system resources used by a container are not allowed to exceed the resource requirements specified in the container definition. Alternatively, if all the resources allocated to a particular resource pool in which the container is executing are not being used, then an application (or service) executing

within the container may use additional resources not specified in the corresponding container definition up to the resources available in the corresponding resource pool. In one embodiment of the invention, a physical memory cap daemon is used to enforce physical memory usage of applications (or services) executing in a container.

[0046] Those skilled in the art will appreciate that while the containers have been described with respect to three resources (*i.e.,* CPU, physical memory, and bandwidth), the invention may be applied to other system resources as well. Thus, the container definition for each container may specify more than the three aforementioned system resources.

[0047] The invention may be implemented on virtually any type of computer regardless of the platform being used. For example, as shown in Figure 5, a networked computer system (200) includes a processor (202), associated memory (204), a storage device (206), and numerous other elements and functionalities typical of today's computers (not shown). The networked computer (200) may also include input means, such as a keyboard (208) and a mouse (210), and output means, such as a monitor (212). The networked computer system (200) is connected to a local area network (LAN) or a wide area network (*e.g.,* the Internet) (not shown) via a network interface connection (not shown). Those skilled in the art will appreciate that these input and output means may take other forms. Further, those skilled in the art will appreciate that one or more elements of the aforementioned computer (200) may be located at a remote location and connected to the other elements over a network. Further, the invention may be implemented on a distributed system having a plurality of nodes, where each portion of the invention (*i.e.,* the helper action, the instrumented application, the tracing framework, etc.) may be located on a different node within the distributed system. In one embodiment of the invention, the node corresponds to a computer system. Alternatively, the node may correspond to a processor with associated physical memory.

[0048] By introducing a container model, only single type of resource management control mechanism (*i.e.,* partitioning) is exposed to the system users. Thus, a single system may be partitioned into a single or multiple resource pools, and each resource

pool may be further partitioned into a single or multiple containers. Internally, the management interface still uses the aforementioned resource management control mechanisms via the management utilities/APIs to achieve the optimal system resource management. Embodiments of the invention provide a method and apparatus to allow a system user to allocate system resources without requiring the system user to understand the individual management utilities/APIs.

[0049] While the invention has been described with respect to a limited number of embodiments, those skilled in the art, having benefit of this disclosure, will appreciate that other embodiments can be devised which do not depart from the scope of the invention as disclosed herein. Accordingly, the scope of the invention should be limited only by the attached claims.